Predicting Gender from Risk-Seeking Behaviors

Deepika Dilip, Rachel Tsong, and Adina Zhang

May 14, 2019

Introduction

One public health phenomenon that has been the subject of debate is the "Gender and Health Paradox", in which women experience decreased quality of life, yet have higher life expectancies than men. A theory that seeks to explain this pattern outlines increased risk-taking by men as a causal mechanism. By this logic, men would indicate more risk-taking behaviors than women when controlled for age and other confounders. We sought to test this theory by examining if risk-seeking behaviors could be predictive of gender.

Young People's Survey

For our analysis, we used a data set consisting of 1,010 responses from a 2013 survey administered among statistics students and their friends at Comenius University in Bratislava, Slovakia. This dataset is available to download from Kaggle. The survey consisted of 150 items including music and movie preferences, hobbies and interests, phobias, health habits, personality traits, views on life, and opinions, spending habits, and demographics. Most of the items were presented as a five-point Likert scale indicating how much a respondent agreed to the statement. Participants were given the survey in both electronic and paper formats. To test our hypotheses, we selected 19 variables that would demonstrate risk-seeking behavior.

After sectioning the data accordingly, we counted missing values to determine if that could affect our results. When only completed responses were considered (i.e. having values for every variable), we had a sample size of 942; 6.7% of the data was missing. As this value was less than 10%, we decided not to address this via imputation and opted for a deletion method (complete-case analysis). The limitations of this method will be addressed in the discussion.

Data Exploration

We checked for collinearity of variables via a correlation matrix; as there were no significant correlations between any predictors, all of our original variables were retained. We stratified responses by gender and visualized them in histograms (fig. 1). Some predictors have more skewed distributions by gender compared to others (e.g. cars, war movies). Other variables such as alcohol use and going outdoors were more evenly distributed by gender.

We used an unsupervised learning technique (principle component analysis) to further explore the data (fig. 2). Due to the large number of predictors, the first and second components only explained 17.1% and 9.4% of the total variance in the data respectively. Nevertheless, PCA was insightful in that some of the variables mostly explained by the first component (interest in cars, adrenaline-driven sports, and active sports) were skewed toward males.

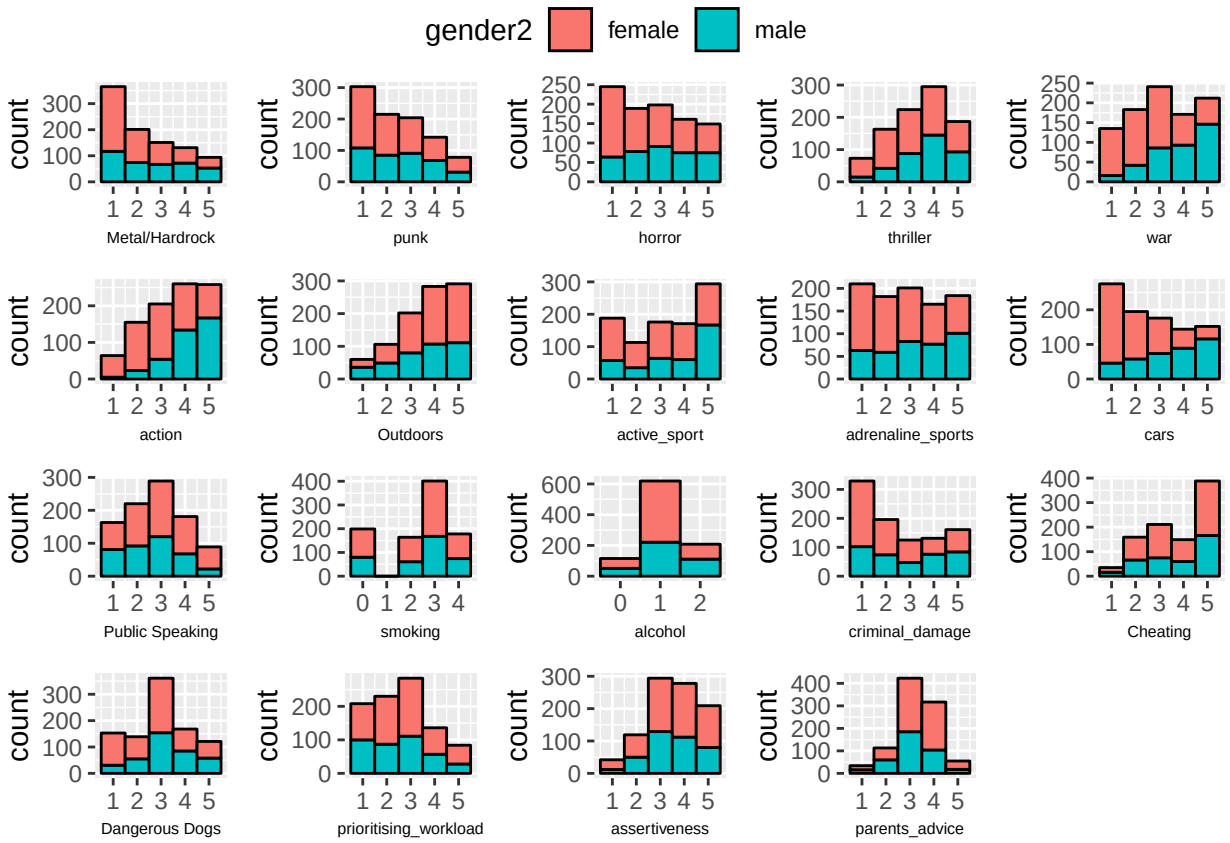

Figure 1: Variable Distributions by Gender

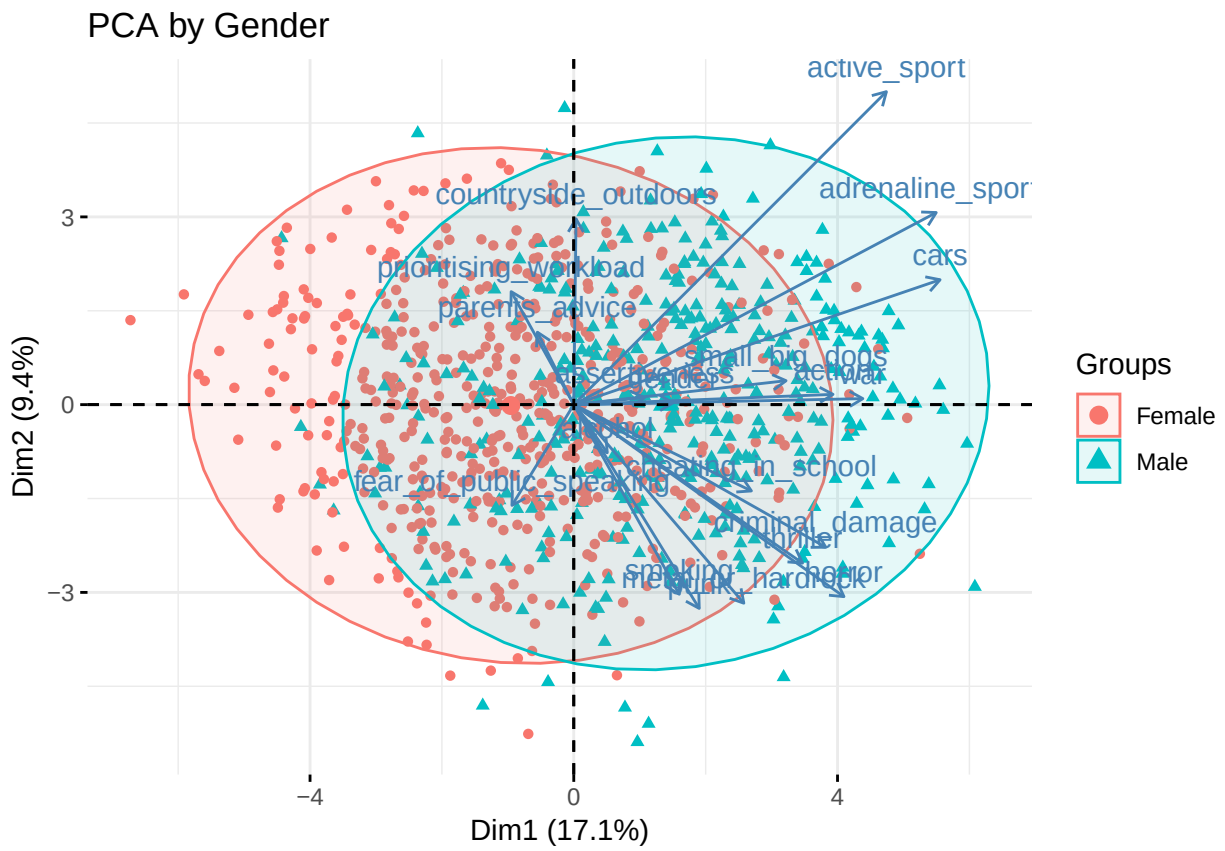

Figure 2: PCA

Method

We selected 19 variables that were related to thrill or risk seeking behaviors. The following variables were included in our analysis:

- Interest in: (ranked 1-5 from "don't enjoy at all" to "enjoy very much")
  - Metal, hard rock music
  - Punk music
  - Horror movies
  - War movies
  - Action movies
  - Outdoor activities
  - Sport at a competitive level
  - Adrenaline sports
  - Cars
- Fears of public speaking (ranked 1-5 from "not afraid at all" to "very afraid of")
- Smoking habits (categorical with 4 levels: "never smoked", "tried smoking", "former smoker", "current smoker")
- Drinking habits (categorical with 3 levels: "never", "social drinker", "drink a lot")
- Statements about personality traits: (ranked 1-5 from "strongly disagree" to "strongly agree")
  - I damaged things in the past when I get angry
  - I used to cheat at school

- I prefer big dangerous dogs to smaller, calmer dogs
- I try to do tasks as soon as possible and not leave them until last minute
- I'm not afraid to give my opinion if I feel strongly about something
- I always listen to my parents' advice

Variables that were ranked on a scale from 1-5 were considered as continous responses. After splitting the data into training and testing sets using an 80/20 split, we fit the following models to the training data:

- Logistic regression
- Discriminant analysis (linear and quadratic)
- KNN
- Support vector machine (linear and radial kernels)
- Classification trees (random forest and boosting)

Training and test error were calculated for each model to evaluate overfitting. The final model was chosen by assessing the training data using the `resamples()` function in the `caret` package.

Models

Logistic Regression

Logistic regression assumes independent observations, a sufficiently large sample size, and a linear relationship between the predictors and the log-odds of the outcome. As we used simple logistic regression, via the `caret` package, it does not contain any tuning parameters. When examining the predictor parameters, the following variables had a p-value of less than 0.001: `cars`, `war`, `action`, and `metal_or_hardrock`, indicating that these are important predictors of gender.

Discriminant Analysis

Linear

For LDA, we used the `caret` package. Linear discriminant analysis assumes normally distributed features. LDA does not contain any tuning parameters and is strong when considering more than 2 predictors. Additionally, it works well when the classes are well separated and Gaussian distribution assumptions are applied. When ranking variables (using the function `varImp()`), the most important ones were as follows: `cars`, `war`, `action`, `active_sport`, and `thriller`.

Quadratic

For QDA, we used the `caret` package. Just like LDA, QDA works well when the classes are well separated and Gaussian distribution (normally -distributed) assumptions are applied. However, unlike LDA, it assumes that each class has its own covariance matrix. QDA does not contain any tuning parameters. When ranking variables (using the function varImp()), the most important ones were as follows: `cars`, `war`, `action`, `active_sport`, and `thriller`.

KNN

The method of K-nearest neighbors makes no assumptions about the shape of the decision boundary; therefore, if the decision boundary is highly non-linear then this model will perform better than others. This model has one tuning parameter, K, the number of closest training points. An optimal value of K was chosen by repeated cross-validation using the `train()` function in the `caret` package by specifying the tuning grid. THe optimal K was 94. One drawback of this model is that we cannot assess variable importance.

SVM

Linear

The support vector classifier has one tuning parameter, C. Using the `train()` function in the `caret` package the best C was 0.0433, which was chosen by cross-validation. Since the support vector classifier is a hyperplane,

the model assumes that the decision boundary is linear, which is a potential drawback if the decision boundary is non-linear.

Radial

The support vector machine with a radial kernel has two tuning parameters. One is C, the cost parameter, and the other is the degree of non-linearity sigma. We used the `train()` function in the `caret` package to choose these parameters by cross-validation. The best C was 78.033 and the best sigma was 0.000335. This model assumes a non-linear boundary. One potential drawback of support vector machines is that you cannot assess variable importance or estimate probabilities as in logistic regression.

Random Forest

A Random Forest (RF) model was fit with the `caret` package using the `ranger` method. We chose Random Forest as one of our models because it is an ensemble method for classification trees that uses a collection of models to improve the final prediction. Random Forest was also chosen over bagging as an esemble method because it minimizes correlation between the decisions trees that are grown. In our model, 500 decision trees were grown and the gini index was selected as a splitrule. The minimum number of variables to randomly sample at each split (mtry) and minimum node size were tuned using repeated cross-validation in the `caret` packages with ROC as the measure of performance. The best tuning results were `mtry = 2` and `min.node.size = 6`. According to the final model, the most important variables for classification are `cars`, `action`, and `war`.

Extreme Gradient Boosting

An Extreme Gradient Boosting (XGB) model was fit using the `xgboost` packages in `caret`. Gradient Boosting is another ensemble method for classification trees. It differs from other ensemble methods such as random forest because the algorithm focuses on modelling errors from previous models in order to improve the bias-variance tradeoff. We selected the `xgboost` package and algorithm because it performs more efficiently than the `gbm` package discussed in class. Multiple parameters were tuned using repeated cross-validation with ROC as the unit of measurement. The final model selected `nrounds = 100`, `max_depth = 2`, `eta = 0.1`, `colsample_bytree = 0.8`. The final model showed that `cars`, `action`, and `war` were the most important variables for classifying gender.

Models	TrainError	TestError
Logistic	0.2066225	0.2032086
LDA	0.2119205	0.1925134
QDA	0.1523179	0.2085561
KNN	0.2291391	0.2139037
SVM Linear	0.2092715	0.1818182
SVM Radial	0.2039735	0.1871658
Random Forest	0.0000000	0.1764706
XGBoost	0.1562914	0.1764706

Table 1: Testing and training error rates among eight classification models

Model Selection

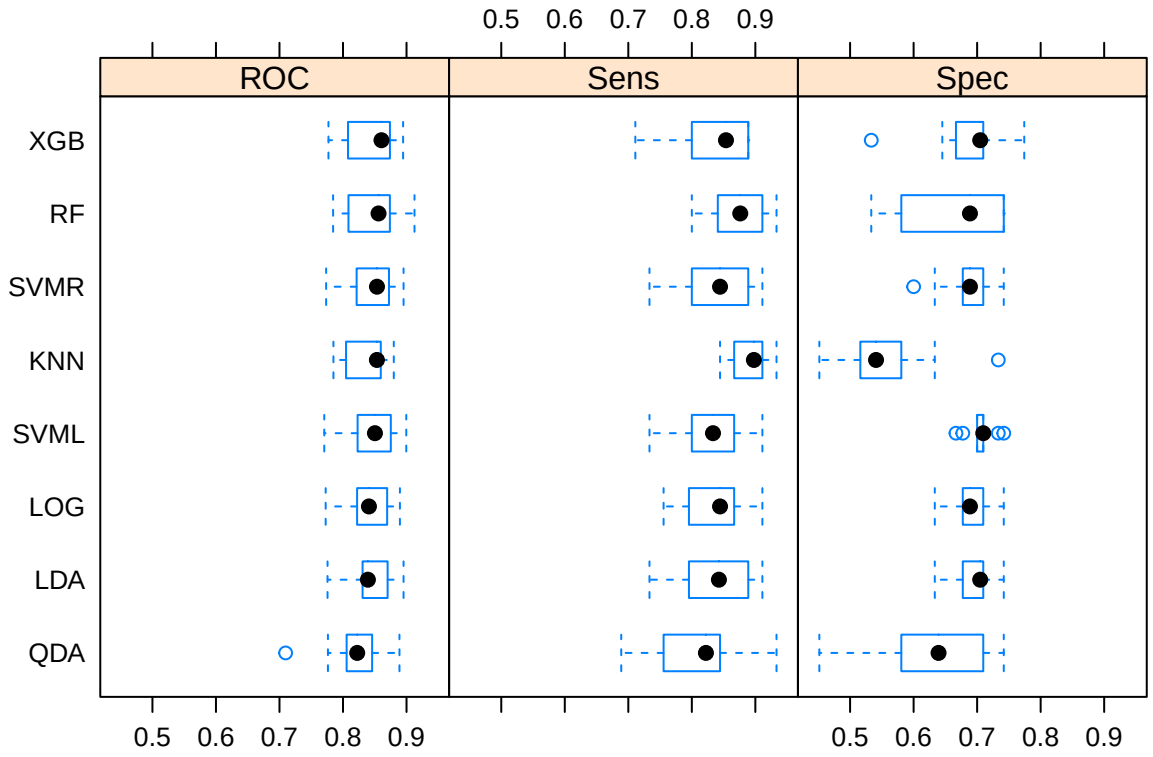

Figure 3: Comparison of Model Fits

Each model was resampled through the `caret` package and the results are shown in the figure above. Overall, all of our models fit well with most ROC's ranging around 80%. The sensitivity and specificity measures have more variability and fall within a wider range. XGBoost and Random Forest perform the best and they perform as well as each other. The median ROC of XGBoost is higher while the means are very similar. The median sensitivity of XGBoost is slightly lower than Random Forest, but the median specificity of XGBoost is higher and the interquartile range is narrower. On the other hand, QDA performed the worst. The median and mean ROC were lowest as was sensitivity. While QDA did not have the lowest specificity, the interquartile range is wide.

Final Model

Based off of our resampling results through the `caret` package, we chose XGBoost as our final model. Althought this model performs as well as random forest, we chose this model because it reported a higher specificity. Furthermore, from Table 1, the Random Forest model seemed to overfit the training data with a training error rate of 1.32%. From XGBoost, the most important variables for classifying individuals by gender were based off their interest in cars, action movies, and war movies (Figure #).

Even though we pick boosting for our a final model, there are limitations to picking this method. A classification tree by itself is very easy to interpret, however the use of ensemble methods utilizes multiple trees making a final model more difficult to visualize or interpret.

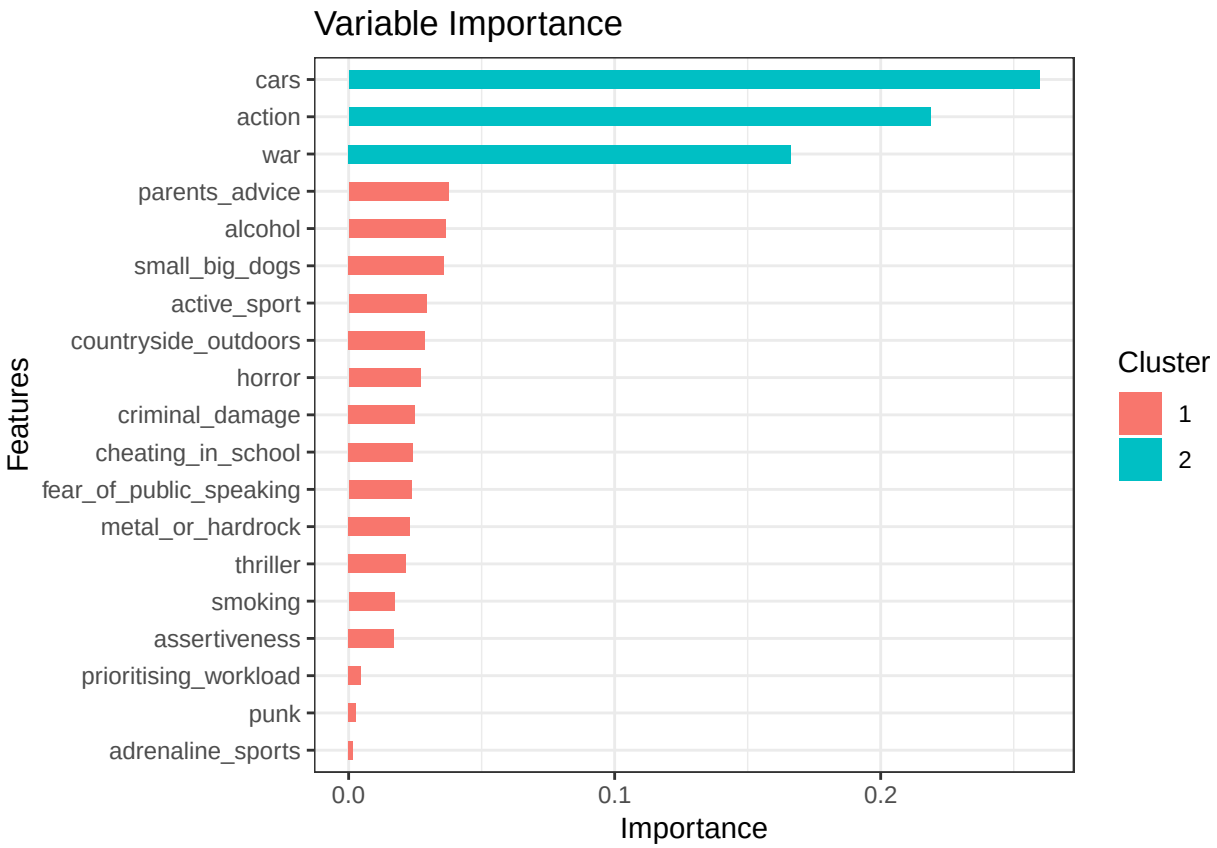

Figure 4: Variable Importance for Final Model

Conclusion

Some of the limitations of this report include translation issues. The original language of the survey was Slovakian, and this could influence interpretations. Additionally, this affects generalizability of findings as the population of this survey was specifically statistics students in Slovakia. The deletion method used would also present issues if data were not missing at random, resulting in bias.

Appendix

Code

```
# libraries
library(tidyverse)
library(dplyr)
library(caret)
library(ranger)
library(xgboost)
library(pROC)
library(factoextra)
library(ggpubr)
library(Ckmeans.1d.dp)

# Load Datset
dat1 = read_csv("./final.csv") %>%
  dplyr::select(-X1) %>%
  mutate(gender = ifelse(gender == 0, "female", "male"))

set.seed(1)
# Partition dataset into training and testing datasets
row_train = createDataPartition(y = dat1$gender, p = 0.8, list = FALSE)
dat1_train = dat1[row_train,]
dat1_test = dat1[-row_train,]

# Set up cross-validation for tuning parameters
ctrl = trainControl(method = "repeatedcv",
                    summaryFunction = twoClassSummary,
                    classProbs = TRUE)

set.seed(1)
# Fit a logistic regression
log.fit = train(gender~., dat1_train,
                method = "glm",
                metric = "ROC",
                trControl = ctrl)

# Fit LDA model
lda.fit = train(gender~., dat1_train,
                  method = "lda",
                  metric = "ROC",
                  trControl = ctrl)
# Fit QDA model
qda.fit <- train(gender~., dat1_train,
                  method = "qda",
                  metric = "ROC",
                  trControl = ctrl)
# fit KNN
knn.fit = train(x = dat1_train[,1:19],
                y = dat1_train$gender,
                method = "knn",
                preProcess = c("center", "scale"),
                tuneGrid = data.frame(k = seq(50, 120, by = 2)),
```

```
                trControl = ctrl)
# Run linear SVM
svm_linear = train(gender ~ . ,
                   data = dat1_train,
                   method = "svmLinear2",
                   preProcess = c("center", "scale"),
                   tuneGrid = data.frame(cost = exp(seq(-5, 1, len = 30))),
                   metric = "ROC",
                   trControl = ctrl)
# Set tuning grid
radial_grid = expand.grid(C = exp(seq(-4, 5, len = 15)),
                          sigma = exp(seq(-8, -5, len = 5)))

# Fit radial SVM model
svm_radial = train(gender ~ .,
                   data = dat1_train,
                   method = "svmRadial",
                   preProcess = c("center", "scale"),
                   tuneGrid = radial_grid,
                   metric = "ROC",
                   trControl = ctrl)
# Set up tuning grid for random forest
rf.grid = expand.grid(mtry = seq(from = 2, to = 6, length.out = 5),
                      splitrule = "gini",
                      min.node.size = 1:10)

# Run Random Forest
rf.fit = train(gender~., dat1_train,
               method = "ranger",
               tuneGrid = rf.grid,
               trControl = ctrl,
               metric = "ROC",
               importance = "impurity")
# set tuning grid
xgbGrid = expand.grid(nrounds = seq(from = 50, to = 200, by = 50),
                      max_depth = c(2, 3, 4, 5, 6),
                      colsample_bytree = seq(0.5, 0.9, length.out = 5),
                      eta = 0.1,
                      gamma = 0,
                      min_child_weight = 1,
                      subsample = 1)

# Run boosting model using xgboost method
xgb.fit = train(gender~., dat1_train,
                trControl = ctrl,
                tuneGrid = xgbGrid,
                method = "xgbTree",
                metric = "ROC",
                importance = "impurity")

# compare models using resamples()
resamp = resamples(list(LOG = log.fit, LDA = lda.fit,
                        QDA = qda.fit, KNN = knn.fit,
```

```
                        SVML = svm_linear, SVMR = svm_radial,
                        RF = rf.fit, XGB = xgb.fit))
bwplot(resamp)
```